

AMENDMENTS TO THE CLAIMS

1. (currently amended) A method comprising the steps of:

providing a graphical debugger interfaced with a model view of a model being executed, said model comprising ~~one or more~~ a blocks ~~where at least one of the one or more blocks~~ that includes a plurality of execution methods, said graphical debugger having debug information related to the execution of said model, said debug information indicating ~~at least one of the~~ an order of ~~the~~ execution of said plurality of execution methods for said ~~at least one of the one or more~~ blocks and a start time or a stop time of ~~at least one of~~ said plurality of execution methods for said block that are executed during the execution of said model; and

outputting said debug information to a user, said debug information allowing the user to determine proper or improper operation for at least a subset of said plurality of execution methods that are executed in said block during the execution of said model.

2. (original) The method of claim 1, comprising the further steps of:

wrapping data generated by the execution of said model in an object, said wrapping encapsulating said execution-generated data in said object; and

exposing said data to said debugger via at least one interface to said object.

3. (original) The method of claim 2, comprising the further step of:

altering said data via said interface.

4. (original) The method of claim 2 wherein said execution-generated data is at least one of state

information, block inputs, block outputs, solver data, profiling data and signal values for said model.

5. (original) The method of claim 1, comprising the further steps of:

processing said model to create compiled model information; and
programmatically generating executable code from said compiled model information,
said code including an interface to said debugger.

6. (previously presented) The method of claim 5, comprising the further step of:

executing said generated code wherein said debugger at least one of sends or receives
information from said executing code during said execution.

7. (previously presented) The method of claim 6, comprising the further steps of:

saving an execution history for said executable code; and
outputting the execution history by at least one of saving it in a permanent memory
location, displaying it for a user, or sending it to a printing device to be printed.

8. (original) The method of claim 6 wherein said debugger is started after compilation and
before the execution of said code.

9. (previously presented) The method of claim 1, comprising the further step of:

indicating graphically with said debugger a plurality of blocks that are part of an
algebraic loop when the executing model is processing the algebraic loop.

10. (previously presented) The method of claim 1, comprising the further step of:

saving a record of a unique execution method invocation, said unique execution method invocation comprising information related to execution of one of said plurality of execution methods that belongs to said ~~at least one of said one or more~~ blocks or to another ~~one of said one or more blocks~~ block in the model, a system, or a model instance in an execution list of called execution methods.

11. (previously presented) The method of claim 10 wherein said unique execution method invocation record comprises information about child records of a subset of said plurality of execution methods executed inside said unique execution method invocation record.

12. (previously presented) The method of claim 11 wherein a link is provided from said unique execution method invocation record to said child record.

13. (previously presented) The method of claim 10 wherein said unique execution method invocation record comprises information regarding at least one parent record of one or more of the plurality of execution methods in which said unique execution method invocation is executed.

14. (previously presented) The method of claim 13 wherein a link is provided from said unique execution method invocation record to said parent record.

15. (previously presented) The method of claim 10 wherein said unique execution method invocation record comprises data about a state of the unique execution method invocation.

16. (previously presented) The method of claim 15 wherein said state indicates the unique execution method invocation is at one of the states of entering, entered, exiting or exited.

17. (previously presented) The method of claim 1, comprising the further step of:
communicating with an external mode simulation using said debugger.

18. (previously presented) The method of claim 1, comprising the further step of:
saving a snapshot of data relating to model execution during execution of said model,
said snapshot data sufficient to enable the subsequent restarting of the execution of said model
using said snapshot data.

19. (previously presented) The method of claim 18 wherein said snapshot data is saved
programmatically at one or more of a regular interval or based on a user-defined parameter.

20. (previously presented) The method of claim 19, comprising the further step of:
loading a saved snapshot into said debugger; and
executing a saved model based on said saved snapshot, said saved model executed from a
point in time said snapshot was saved using information from said saved snapshot.

21. (original) The method of claim 18, comprising the further step of:

displaying graphically to a user the saved snapshot data.

22. (original) The method of claim 21, comprising the further step of:

displaying graphically to a user at least one additional set of snapshot data without restarting the execution of said model.

23. (previously presented) The method of claim 22 wherein said set of snapshot data is displayed in order of decreasing time.

24. (original) The method of claim 18, comprising the further step of:

saving a difference between a set of current model execution data and a saved snapshot.

25. (currently amended) A medium for use in a modeling and execution environment on an electronic device, said medium holding executable instructions on the electronic device for performing an execution method, said method comprising the steps of:

providing a graphical debugger interfaced with a model view of a model being executed, said model comprising a block that includes a plurality of execution methods, said graphical debugger having debug information related to the execution of said model, said debug information indicating at least one of the order of the execution of a plurality of execution methods in said model and a start time or a stop time of at least one execution method executed during the execution of said model; and

outputting said debug information to a user, said debug information allowing the user to determine proper or improper operation for at least a subset of said plurality of execution

methods for the block that are executed during the execution of said model.

26. (original) The medium of claim 25, wherein said method comprises the further steps of:

wrapping data generated by the execution of said model in an object, said wrapping encapsulating said execution-generated data in said object; and
exposing said data to said debugger via at least one interface to said object.

27. (previously presented) The method of claim 26, comprising the further step of:

altering said data via said at least one interface.

28. (previously presented) The medium of claim 26 wherein said execution-generated data is at least one of state information, block inputs, block outputs, or signal values for said model.

29. (previously presented) The medium of claim 25, wherein said method comprises the further steps of:

processing said model to create compiled model information; and
programmatically generating executable code from said compiled model information,
said generated code including an interface to said debugger.

30. (original) The medium of claim 29, wherein said method comprises the further step of:

executing said generated code wherein said debugger sends and receives information from said executing code during said execution.

31. (previously presented) The medium of claim 30 wherein said method comprises the further steps of:

saving an execution history for said executed code; and

outputting the execution history by at least one of saving it in a permanent memory location, displaying it for a user, or sending it to a printing device to be printed.

32. (previously presented) The medium of claim 30, comprising the further steps of:

initiating said executing; and

starting said debugger subsequent to said initiating.

33. (original) The medium of claim 25, wherein said method comprises the further step of:

indicating graphically with said debugger a plurality of blocks that are part of an algebraic loop when execution of the model is processing the algebraic loop.

34. (previously presented) The medium of claim 25, wherein said method comprises the further step of:

saving a record of a unique execution method invocation, said unique execution method invocation comprising information related to the execution of an execution method that belongs to at least one of a block, a system, or a model instance in an execution list of called execution methods.

35. (previously presented) The medium of claim 34 wherein said unique execution method invocation record comprises information about child records of execution methods executed

inside the unique execution method invocation record.

36. (previously presented) The medium of claim 35 wherein a link is provided from said unique execution method invocation record to said child record.

37. (previously presented) The medium of claim 34 wherein said unique execution method invocation record includes information regarding at least one parent record of execution methods in which the unique execution method invocation is executed.

38. (previously presented) The medium of claim 37 wherein a link is provided from said unique execution method invocation record to said parent record.

39. (previously presented) The medium of claim 34 wherein said unique execution method invocation record comprises data about the state of the execution method invocation.

40. (previously presented) The medium of claim 39 wherein said state indicates the execution method invocation is at one of the states of entering, entered, exiting or exited.

41. (original) The medium of claim 25, wherein said method comprises the further step of: communicating with an external mode simulation with said debugger.

42. (previously presented) The medium of claim 25, wherein said method comprises the further step of:

saving a snapshot of data relating to model execution during execution of said model, said snapshot data sufficient to enable subsequent restarting of the execution of said model using said snapshot data at a saved point in time.

43. (previously presented) The medium of claim 42 wherein said snapshot data is saved programmatically at one or more of a regular or user-defined interval.

44. (previously presented) The medium of claim 42, wherein said method comprises the further step of:

loading a saved snapshot into said debugger; and
executing a saved model from the point in time said snapshot was saved.

45. (original) The medium of claim 42 wherein said method comprises the further step of:
displaying graphically to a user the saved snapshot data.

46. (original) The medium of claim 45 wherein said method comprises the further step of:
displaying graphically to a user at least one additional set of snapshot data without restarting the execution of said model.

47. (previously presented) The medium of claim 46 wherein said set of snapshot data is displayed in order of decreasing time.

48. (currently amended) A computer-implemented method, comprising:

identifying a first execution method operating in a first ~~domain~~ environment of a computer-based modeling application that executes a ~~computer-based~~ model, where the first ~~domain~~ environment is one of a text-based environment, a time-based block diagram, a state based block diagram, or a data-flow diagram;

identifying a second execution method operating in a second ~~domain~~ environment, where the second ~~domain~~ environment differs from the first ~~domain~~ environment;

debugging the first execution method and the second execution method while the computer-based model operates on behalf of a user; and

generating output information for the user or for a destination, the output information identifying when the first execution method or the second execution method are operating, identifying an operation performed by the first execution method or the second execution method at a determined location in the first execution method or the second execution method, or identifying an error related to the first execution method or the second execution method during execution of the computer-based model.

49. (previously presented) The method of claim 48, further comprising:

displaying the output information to a user or sending the output information to the destination.

50. (previously presented) The method of claim 48, further comprising:

providing a destination interface, the destination interface allowing extensible debugging of the first execution method and another execution method or debugging of the second execution method and the another execution method.

51. (previously presented) The method of claim 48, further comprising:

displaying a hierarchy containing information about the first execution method or the second execution method, the hierarchy allowing a user to identify relationships between the first execution method and the second execution method, the first execution method and another execution method, or the second execution method and the another execution method.

52. (previously presented) The method of claim 48, wherein the debugging further comprises:

debugging the first execution method or the second execution method with respect to a block diagram.

53. (previously presented) The method of claim 48, further comprising:

identifying the first execution method or the second execution method using a visual indicator to identify when the first execution method or the second execution method is executing.

54. (previously presented) A method, comprising:

receiving information about a first execution method and a second execution method on behalf of a graphical model comprising blocks, where at least one of the blocks includes the first execution method and at least one other execution method or the second execution method and the at least one other execution method, where the first execution method or the second execution method are related to one or more of the blocks;

identifying at least a portion of the first execution method or the second execution

method when the first execution method or the second execution method are running, respectively;

obtaining information about the running of the first execution method or the second execution method using the identifying; and

providing debugging information to a user via a display or providing debugging information to a destination device, the debugging information identifying the first execution method or the second execution method and information about the first execution method or the second execution method, respectively.

55. (previously presented) A method, comprising:

identifying a first root method comprising one or more child methods, the first root method related to a graphical modeling application;

identifying a second root method related to the graphical modeling application;

running the first root method and the second root method in a graphical debugger to obtain information about the operation of the first root method or the second root method; and

displaying a debugging result to a destination, the debugging result comprising visual identifiers related to the operation of the first root method, the one or more child methods or the second root method, error information about the first root method, the one or more child methods or the second root method, an execution result for the first root method, the one or more child methods or the second root method, or status information related to the first root method, the one or more child methods or the second root method.

56. (previously presented) A method for implementing a user interface for debugging a graphical

model, the method comprising:

displaying a hierarchy comprising information about a first root method, one or more child methods related to the first root method, or a second root method, the hierarchy displaying information about the first root method, the one or more child methods, or the second root method in an arrangement representing a relationship among the first root method, the one or more child methods, or the second root method; and

displaying an indicator on the hierarchy proximate to the first root method, the one or more child methods, or the second root method, the indicator denoting a status of the first root method, the one or more child methods, or the second root method, where the status indicates whether the first root method, the one or more child methods, or the second root method are operating according to determined parameters.

57. (previously presented) The method of claim 56, wherein the displaying an indicator further comprises:

displaying a first symbol when the status is related to the first root method; and

displaying a second symbol when the status is related to the one or more child methods or the second root method.

58. (previously presented) The method of claim 56, wherein the displaying an indicator further comprises:

displaying a first color to represent a first status related to the first root method; and

displaying a second color to represent a second status related to one of the one or more child methods or the second root method.

59. (previously presented) The method of claim 56, further comprising:

displaying the hierarchy in a first region related to one or more display devices; and
displaying a graphical diagram related to the first root method or the second root method
in a second region related to the one or more display devices, the graphical diagram
synchronized with information displayed in the first region.

60. (previously presented) The method of claim 56, further comprising:

displaying a first indicator in a first region proximate to the first root method, the one or
more child methods, or the second root method; and
displaying a second indicator in a second region, where the first indicator and the second
indicator identify a relationship between information displayed in the first region and
information displayed in the second region.

61. (previously presented) A method for debugging operation of a graphical icon, the method
comprising:

identifying a plurality of execution methods for the graphical icon using a plurality of
regions related to the graphical icon;

displaying information about a first one of the plurality of execution methods in a first
one of the plurality of regions or information about a second one of the plurality of execution
methods in a second one of the plurality of regions; and

associating the information in the first one of the plurality of regions or information in the
second one of the plurality of regions with a graphical debugger to provide a user with

debugging results for the first one of the plurality of execution methods or the second one of the plurality of execution methods, the debugging results allowing the user to identify desirable operations performed on behalf of the graphical icon or undesirable operations performed on behalf of the graphical icon.

62. (previously presented) The method of claim 61, further comprising:

- displaying information about a first one of the plurality of execution methods in a first region of a display area; and

- displaying the debugging results in a second region of the display area, the displayed debugging results linked to at least a portion of the displayed information in the first region.

63. (previously presented) The method of claim 61, further comprising:

- displaying a hierarchy in a region of the display device; and

- displaying the debugging results for the first one of the plurality of execution methods or the second one of the plurality of execution methods using the hierarchy.

64. (previously presented) The method of claim 61, wherein the first one of the plurality of execution methods is a root method comprising one or more child methods, and wherein the method further comprises:

- displaying the graphical icon and graphical icon debugging information in a first display area, the graphical icon debugging information mapped to the graphical icon; and

- displaying information about the root method and the one or more child methods in a hierarchy, where information in the hierarchy is linked to the graphical icon debugging

information.

65. (previously presented) The method of claim 64, further comprising:

identifying the root method or one or more of the one or more child methods using a first indicator; and

identifying the graphical icon debugging information using a second indicator.

66. (previously presented) The method of claim 65, wherein the first indicator or the second indicator are a color, a pointer, a symbol, a font, or a border.

67. (previously presented) The method of claim 64, wherein the first display area comprises a window that displays information about the graphical icon or the graphical icon debugging information.

68. (previously presented) The method of claim 67, wherein the window comprises a visual indicator to connect the window to the graphical icon or to the graphical icon debugging information.

69. (previously presented) The method of claim 64, further comprising:

displaying an execution list in the hierarchy, the execution list related to the root method or the one or more child methods.

70. (new) The method of claim 1, wherein the model comprises a plurality of blocks having

execution methods, and wherein the debug information indicates an order of execution of said execution methods of said plurality of blocks, during execution of the model.

71. (new) The medium of claim 25, wherein the model comprises a plurality of blocks having execution methods, and wherein the debug information indicates an order of execution of said execution methods of said plurality of blocks, during execution of the model